49장 Babel과 Webpack을 이용한 ES6+/ES.NEXT 개발 환경 구축

💡 **ES6+**
ES6 이상의 버전

ES.NEXT
ES 제안 사양(출시 예정 또는 실험적인)

- 대부분 브라우저의 ES6(ES2015) 지원율은 98%인데 반해 IE 11의 지원율은 약 11%이며,

  매년 도입되는 ES6+와 ES.NEXT는 브라우저에 따라 지원율이 제각각
  - 따라서 최신 ECMAScript 사양을 사용하여 프로젝트를 진행하려면

    최신 사양으로 작성된 코드를 경우에 따라 IE를 포함한 구형 브라우저에서

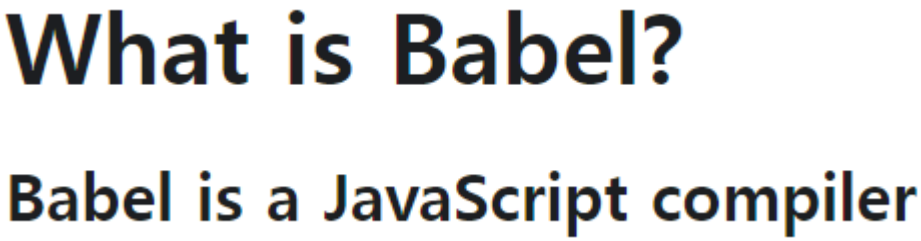
    문제 없이 동작시키기 위한 개발 환경을 구축하는 것이 필요하다.
  - 또한 대부분 프로젝트에서 모듈을 사용하기에 모듈 로더도 필요하다.
    - ES6 모듈(ESM)은 대부분의 모던 브라우저에서 사용할 수 있으나, 다음의 이유로 아직까지는 별도의 모듈 로더를 사용한다.
      - IE를 포함한 구형 브라우저는 ESM을 지원하지 않다.
      - ESM을 사용하더라도 여전히 트랜스파일링이나 번들링이 필요하다.
      - ESM이 아직 지원하지 않는 기능이 있으며, 해결되고 있으나 아직 이슈가 존재한다.

프로젝트 목표

> 1. 트랜스파일러인 Babel과 모듈 번들러인 WebPack을 이용하여 ES6+/ES.NEXT 개발 환경 구축

> 2. Webpack을 이용하여 Babel을 로드하여 ES6+/ES.NEXT 사양의 소스코드를
> 구형 브라우저에서도 동작하도록 ES5 사양의 소스코드로 트랜스파일링

49.1 Babel



https://babeljs.io/docs/

49.1.1 정의

Babel은 주로 ECMAScript2015+ 코드를 현재 및 이전 브라우저나 환경에서 이전 버전과 호환되는 JavaScript 버전으로 변환하는 데 사용되는 툴체인입니다.

> 💡 **자바스크립트는 인터프리터 언어라고 했는데..?**
> 정확히는 컴파일 언어라기 보다 최신 자바스크립트 코드를 구형 자바스크립트 코드로
> **변환**하는 **트랜스파일러**가 정확한 표현!!

49.1.2 주요 기능

- Transform syntax (구문 변환)

  트랜스파일링으로 최신의 자바스크립트 문법을 오래된 브라우저가 이해할 수 있도록 구형 문법으로 변환

- Babel-polyfill을 통해서 폴리필 기능을 지원

  폴리필 : 오래된 브라우저에 네이티브로 지원하지 않는, 사용자가 사용하는 메서드·속성·API가

  존재하지 않을 때 추가

  바벨은 트랜스파일러 역할만 할 뿐, 최신 함수를 사용할 수 있는 것은 아니다.

  폴리필은 프로그램이 처음 시작할 때 지원하지 않는 기능들을 지원해 준다.

  ✅ 바벨은 컴파일 때 실행되고 폴리필은 런타임에 실행된다.

  > 💡 **폴리필이 런타임에 실행되는 이유?**
  > 브라우저가 특정 기능을 지원하지 않는 경우, 해당 기능을 구현한 폴리필 실행

- JSX and React

  바벨은 JSX 문법을 변환한다.

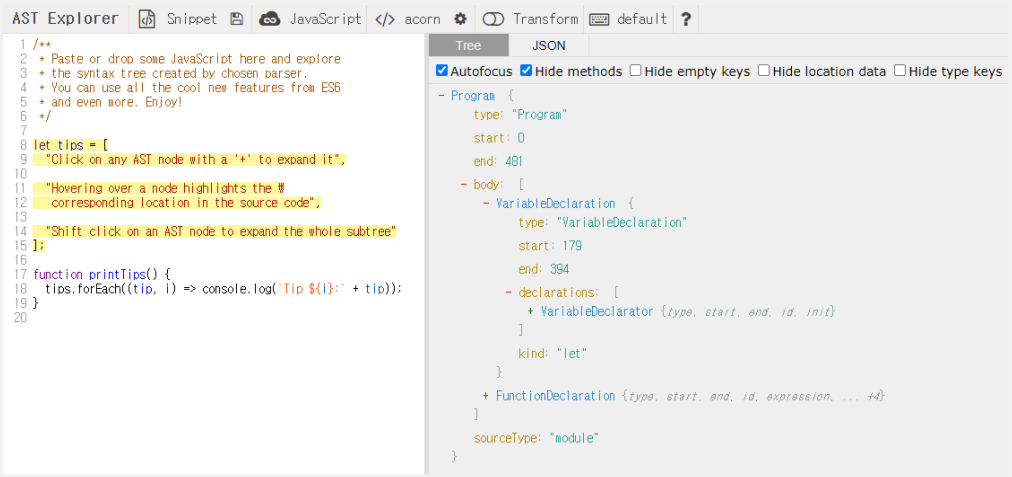
  > 💡 **JSX(**JavaScript + xml)
  > 템플릿 언어(HTML 내부에 변수 및 문법을 사용할 수 있는 언어)로 보일 수 있으나,
  >
  > **JavaScript 코드 내부**에서 **HTML 문법을 사용**한 것

49.1.3 동작 과정

1. **Parsing (파싱)**

   소스 코드를 분석하여 AST로 변환

💡 **AST(Abstract Syntax Tree)**
소스 코드의 추상 구문 구조의 트리
컴파일러에서
자료 구조로 사용되며 컴파일러의 구문 부석(parsing) 단계의 트리로 표현된 결과



AST 변환 결과 확인 사이트

2. **Transformation (변환)**

   @babel/traverse를 사용하여 변환된 AST를 순회하며 각 AST 노드들은 브라우저가 지원하는 코드를 나타내는 새로운 노드들로 대체되고 새로운 AST로 변경

3. **Code Generation (코드 생성)**

   새로운 AST를 바탕으로 @babel/generator를 통하여 새로운 코드를 생성

49.2 Webpack

1. 정의

webpack은 모던 JavaScript 애플리케이션을 위한 **정적 모듈 번들러**입니다.

> 💡 **모듈 번들러?**
>
> 웹 애플리케이션을 구성하는 자원(HTML, CSS, JavaScript, Image 등)을 모두 각각의 모듈로
> 보고 이를 조합해서 병합된 하나의 결과물을 만드는 도구

⇒ 여러 파일을 하나 이상의 파일로 합쳐주는 **자바스크립트 번들러**!!

2. 주요 기능

- **성능 최적화 & 자동화**

  코드 축소와 더불어 사용하지 않는 코드를 제거하는 tree shaking과 같은 최적화를 수행함으로써
  HTTP 요청 수를 감소하여 웹사이트 성능을 궁극적으로 향상시키고, 로딩속도를 빠르게 향상

- **파일 단위의 자바스크립트 모듈 관리**

  HTML, CSS, JavaScript, Images, Font 등 모든 파일 하나 하나 나누어 모듈화하여, 웹 애플리케이션을 구성

- **번들러가 제공하는 편의성**

  CSS가 아닌 Sass나 Stylus 등을 사용할 경우, 또는 TypeScript 사용 시 번들러가 컴파일 과정에서 필요한 플러그인을 추가하고 번들러를 실행

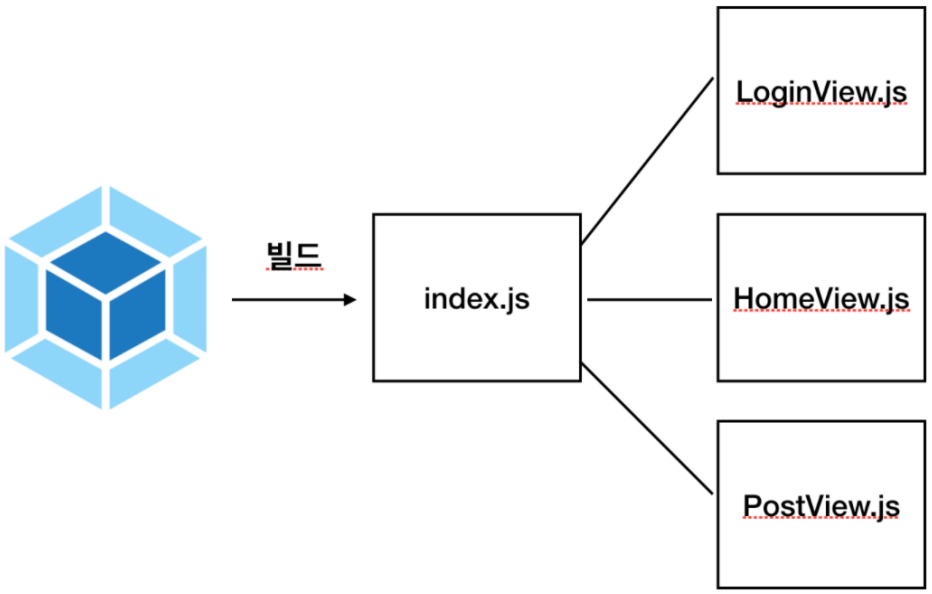
- **Dependency Issue(종속성 문제) 해결**

  서버와 브라우저 모두에서 최대한 원활하게 작동할 수 있는 코드의 상당부분을 빌드 시 모든 종속성과 함께 번들하는데 도움을 준다.

3. 핵심 요소

- **Entry**
  - 웹팩에서 웹 자원을 변환하기 위한 최초 진입점

- entry로 묶고자하는 파일의 첫 번째 진입점을 설정
- 엔트리 설정 후 웹팩을 실행하면 아래와 같이 파일이 빌드된다.



디펜던시 그래프

- **Output**
  - 웹팩을 실행하여 빌드하고 난 후 결과물의 파일 경로
  - filename : 웹팩으로 빌드한 파일의 이름
  - path : 해당 파일의 경로

- **Loader**
  - 웹팩이 애플리케이션을 해석할 때 자바스크립트 파일이 아닌 HTML, CSS, Images, font 등을 변환할 수 있게 도와주는 속성
  - 웹팩은 모든 파일을 모듈로 취급하여 관리하지만, 사실상 자바스크립트 파일만 알고 있기 때문에 로더를 이용해 다른 파일들을 웹팩이 이해할 수 있게 변경해야 한다.

> !! 로더로 설정을 지정해주지 않으면 웹팩이 해당 파일을 읽을 수 없기 때문에 에러 발생!

- **Plugin**

- 웹팩의 기본적인 동작에 추가적인 기능 제공
- 해당 결과물의 형태를 바꾸는 역할(ex.웹팩으로 빌드한 결과물로 HTML 파일을 생성)

- **Mode**
  - 웹팩에 내장된 환경별 최적화 활성화
  - development, production, none(기본값은 production)

```
// webpack.config.js
const path = require('path');

module.exports = {
        // Entry (엔트리)
    entry: ['@babel/polyfill', './src/js/main.js'],

    // Output (출력)
    output: {
        path: path.resolve(__dirname, 'dist'),
        filename: 'js/bundle.js'
    },

    // Loaders (로더)
    module: {
        rules: [
            {
                test: /\.js$/,
                include: [
                    path.resolve(__dirname, 'src/js')
                ],
                exclude: /node_modules/,
                use: {
                    loader: 'babel-loader',
                    options: {
                        presets: ['@babel/preset-env'],
                        plugins: ['@babel/plugin-proposal-cla
                    }
```

```
                }
            }
        ]
    },
    devtool: 'source-map',

    // Mode (모드)
    mode: 'development'
}
```